Edinburgh Napier University

SET08101 Web Tech

Lab 2 - HTML

Dr Simon Wells

1 Aims

At the end of the practical portion of this topic you will:

-

NOTICE: You should be putting your lab work into Git (Add, Commit, Push) as you develop it. This is active & deliberate practise and means that you are constantly building skills so that you no longer have to waste cognitive energy on then; you just use them. If you are looking for a good Git tool then Git-Bash is probably the best and available from here: `https://git-scm.com/download/win`**. Rather then the defaul that tries to auto-download, I'd choose the portable "thumbdrive" edition. You can then follow this tutorial to use git-bash with GitHub:** `http://vastinfos.com/2016/09/quick-github-and-gitbash-basics-for-beginners-tutorial/`

2 Activities

2.1 HTML Documents

We saw a simple HTML document last week that looked like this:

```
<html>
    <head>
        <title>SET08101 - Web Tech</title>
    </head>
    <body>
        <h1>Hello Web Tech</h1>
    </body>
<html>
```

We also discovered that the browser is very good at fixing problems in the supplied document. The DOM which is parsed from the HTML file is an internal model that the browser builds of how the HTML is strucured. It actually expects a DOCTYPE tag to begin things so our example last week wasn't strictly correct. It should have been like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Web Tech</title>
    </head>
    <body>
        <h1>Hello Web Tech</h1>
    </body>
<html>
```

We can simplify that some more but it doesn't look very interesting in the browser because we can remove the head, body, title and heading tags and their associated content to give a minimal valid HTML document as follows:

```
<!DOCTYPE html>
<html>
<html>
```

2.2 HTML Elements

It is from this minimal set of tags in the HTML document example above that all possible valid HTML documents are built. One of the core ideas in HTML is that it is a tree structure. It starts with a root tag pair, the HTML tags, which enclose other pairs of tags, usually at this stage the head and body tags, which in turn enclose other pairs of tags, until you're done, i.e. you've described all of the text that you want the document to hold. These tag paris are called Elements. Each element has a start tag and an end tag, for example, a start tag might look like this: <title> and an end tag might look like this: </title>. Notice that the sole difference between the two tags is a single backslash character.

Note that elements usually have content between start and end tag. If an element has no content then it is called an empty element. Empty elements do not have an end tag, because they are not enclosing anything. An example of this is the line break element <br> which stands on its own. Many people include the closing element when using empty tags, e.g. <br />. One this we should notice is that browsers are quite generous in what they will accept as valid HTML. This is both a blessing and a curse. A blessing because it means that lots of people can write nearly correct HTML and thus create their own websites. This means that there is a very low barrier to entry to publishing on the web which has arguably been to its advantage. However, it means that browsers have had to be a bit creative about how they render supplied HTML. This means that historically, it has been difficult to get a designers vision to render in exactly the sane way across multiple browsers because each has made it's own decision about how to treat the elements that make up HTML.

2.3 HTML Attributes

Attributes are additional "settings" for an attribute, they enable you to specify how it behaves or looks. All HTML elements have attributes which provide additional information about that element. Attributes are always specified in the start tag and always come in name/value pairs, e.g. name="value". For every HTML element that we discover there will usually be many attributes available and you should explore the range of attributes available. As you discover new HTML elements over the remainder of the trimester, make sure to look up the element on the Mozilla Developer Reference Pages[1] to see what attributes are available.

2.4 Basic HTML Elements

Some of the most basic elements relate to core text markup. Methods for indicating what role a given piece of text plays in the wider document. Just like writing a word processor document, where we use headings, paragraphs, etc. We do the same in HTML. These basic text markup elements are generally used between the <body> tags of the document.

The heading tags are defined for six levels of heading, ranging from <h1> the most important heading, to <h6>, the least important, e.g.

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Headings</title>
    </head>
    <body>
        <h1>A Very Important Heading</h1>
    </body>
<html>
```

- Explore the other headings that are available and see how they look in the browser
- Make an HTML document that shows all the heading elements on one page.

Paragraphs are a basic building block for organising writing within a larger textual document so there is also a paragraph element using the <p> tags.

- Make an HTML document that incorporates several paragraphs of text. You can select some text from a news website, or better yet, investigate "greeking" or "lorem ipsum", a way to generate blocks of text for testing websites before the 'copy' is written. There are many websites that generate lorem ipsum text as a service for you to copy and paste into website designs whilst you are waiting for the content to be written. This can be very useful.

We probably want to have some more visual appeal that just using text in our html document. We can include images very easily using the <img> tag, e.g.

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Headings</title>
```

[1]https://developer.mozilla.org/en-US/docs/Web/HTML/Element

```
    </head>
    <body>
        <img src="https://raw.githubusercontent.com/siwells/set08101/master/
            resources/vmask.jpg" />
    </body>
<html>
```

We can either use a link to an image from a website on the web, e.g. `https://raw.githubusercontent.com/siwells/set08101/master/resources/vmask.jpg` as in the example above, or we can use a local image file. If you do use a link to an image that is someone else's site then this is often considered bad form. You also have no control over what your link actually points to and the image could be replaced with something else at any point. The best case scenario is that the image link becomes broken.

Download the vmask image above, or use another of your choice and put it in the same folder as your HTML file. Rather than a "fully qualified domain name" like we had above, we can now just use the local filename. e.g.

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Headings</title>
    </head>
    <body>
        <img src="vmask.jpg" />
    </body>
<html>
```

Note that you can place image into a subfolder but you will have to specify the path to that folder in your HTML, e.g. if the image is in the images sub-folder then:

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Headings</title>
    </head>
    <body>
        <img src="images/vmask.jpg" />
    </body>
<html>
```

- Try out some different ways to organise your images using basic HTML layouts elements. You might want to investigate, for example, the table element.

There are many HTML elements, too many to list here, and too many to adequately explore in a single lab. Instead, explore the list of elements on the Mozilla Developer Reference Pages[2] and get a feel for the range of elements that can be used to mark up an HTML document.

2.5 HyperText

Arguably the most important element in HTML are the tags that support hypertext. These enable links between pages. They put the "Hyper" into HTML which would otherwise be TexMarkupLanguage (TML) and are also responsible for putting the "web" aspect into the World Wide Web.

Links are descriped using the <a> tag which encloses a section of text, or some other element. The href, or hypertext reference, attribute is then used to specify a Uniform Resource Locator (URL) or "web address" which indicates where the link points to. For example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - HTML Links</title>
    </head>
```

[2] `https://developer.mozilla.org/en-US/docs/Web/HTML/Element`

```
    <body>
        <a href="https://siwells.github.io/set08101/">Module Website</a>
    </body>
<html>
```

Will create a piece of text in the body of our html document which links to our module webpage. Load it up in a browser and click the link. It should take you to our module webpage. Try adding another link to another website of your choice. What other tags might you need to use to arrange these links in a nice way. As we design our web sites we must consider how and when we want a link to work, sometimes we might want to list a collection of links, but at other times we might want to include our links inline in our paragraphs of text.

- Create an HTML document that contains a list of links to your favourite 5 websites (if you don't have 5 favourite websites then just find 5 useful links that you can use)
- Create an HTML document that contains the following paragraph of text:

  Web Tech is a module that is run in the School of Computing at Edinburgh Napier University.

  Turn the 'Web Tech", "School of Computing", and "Edinburgh Napier University" phrases into appropriate links.

You can also make many other HTML elements into links. In a sense, Hypertext is not just text but other media, for example, images. So the idea of Hypertext gives way to a more general sense of Hypermedia. You can make a clickable image by enclosing an image element in <a> tags.

- Create an HTML document that contains an image and make it into a clickable link
- Explore other HTML elements to see which can be made into hyperlinks (you might have to do some background reading at this point [and the situation might be dependent upon the browser you're using].

You can also create links between local pages, i.e. pages on the same site. For our purposes now we'll consider local to mean within the same folder hierarchy.

- Create a folder and add an index.html to it. Add some further html files to the folder with different names, e.g. 01.html, 02.html, etc. Now add text and links within each file to enable you to open the index.html and navigate to the other files. Consider how you might navigate back to the index.html.

Links provide not only the basis for Hypertext but enable us to break down our own site into separate pages and navigate between them. For this reason they are possibly one of the most important tags available in HTML. We can do without styles, or ways differentiate a paragraph from a heading, but we can't do without hyper-links.

2.6 Challenge

Create a set of simple and short web pages to describe yourself and your career at Napier. If that sounds boring you can choose another topic, perhaps a hobby or something else that you are interested in. Where appropriate you should include links to external resources, e.g. to various pages in the Napier website. You should be considering how to break things down into separate pages, and also what internal links you might need for navigation within each page and between your pages. For example, starting within a homepage, i.e. index.html you might have links to a personal page about you, and a course page about your degree. Your course page might be broken down by year, and each year in turn by module. At each point you should be considering the best HTML elements to apply to markup your page. Don't worry too much about how things look at this point, next week we'll look at prettifying things using CSS, what we want for now is nicely organised information that we can navigate in the browser.

2.7 Finally

If you want some additional practise then the W3Schools HTML exercises are a great place to start:

- HTML Exercises: `https://www.w3schools.com/html/exercise.asp`